Universiteit van Amsterdam

A Feasibility Study in Unsupervised Estimation of a Stochastic CFG for Latin Morphology

MA Thesis

Kelly Nedwick

June 11, 2004

CONTENTS

LIST OF ILLUSTRATIONS

Abstract

This report describes the development of a hybrid system for morphological analysis of classical Latin and two experiments. The system first isolates the morphological ending of each word in a sentence using simple deterministic methods. A probabilistic module then uses the Inside-Outside algorithm to determine the most likely POS tag for the endings. An examination of the procedure and results of these preliminary experiments provides solid ground for continuing to research this sort of hybrid system.

Introduction

Since the Turing Test was developed in the 1950s computers and linguistics have been intertwined. Of course, in all this time, relatively few linguists have taken the time to learn complex computational concepts and few computer scientists really had a good understanding of linguistic motivations and impulses.[1] In the 1990s however, advances in computer technology and the field of statistics led to a boom in natural language processing enjoyed by linguists and computer scientists alike.

Today, computers are used for everything from generating weather reports to translation to machine grading of school papers. By this point everyone has dealt with an automated phone system or taken a computer-based test like the GRE. Although amazing progress has been made in the field of NLP, there are still some hurdles to be overcome.

As the title suggests the following text deals with the development of a statistical morphological analyzer for classical Latin. Classical Latin has a rich morphology meaning that it used morphological suffixes to convey syntactic information. For each morph there is often more than one syntactic category to which it could possibly belong. While a simple morphological analyzer may be called upon to return all possibilities, this analyzer has been challenged to disambiguate between the many possible meanings for a single Latin morph. This involved the creation of two modules, a deterministic preprocessor and a probabilistic postprocessor as well as a Latin corpus and suitable Latin grammar. The following gives the linguistic and technological motivations for its unique structure, the work involved in its development as

[1] This is a rather general statement. Until recently, the computer skills needed to handle NLP were formidable. Also the popularity of Chomsky's Generative Grammar did nothing to encourage linguists to develop computational tools or methods.

well as the results of two experiments designed to test the system's feasibility.

As a taste of what's to come, figure 1 below shows the system in its entirety, giving a rough description of how the cascade model works. The deterministic preprocessor, a series of Perl programs, takes in

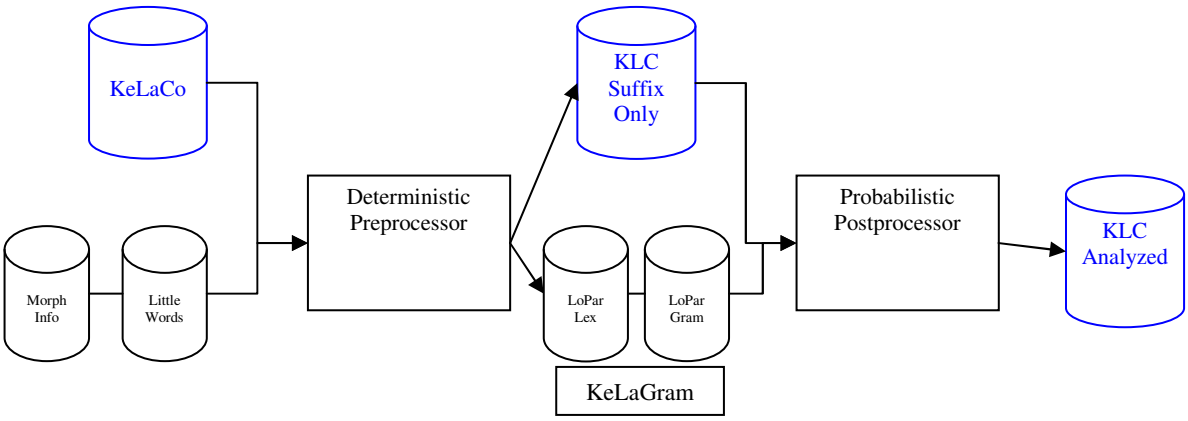


Figure 1 The Big Picture
The hybrid system consisting of a deterministic and a probabilistic module transforms the input to analyze the corpus.

two files of information and the unformatted Latin Corpus KeLaCo and returns the suffix only version of KeLaCo as well as the LoPar grammar and lexicon files which make up KeLaGram, a CFG. These files are used as the input to the probabilistic postprocessor which returns KeLaCo complete with the morphological tags and KeLaGram the PCFG with rule probabilities as estimated by the Inside-Outside algorithm.

In this hybrid system, the probabilistic postprocessor uses statistics to reduce ambiguity and select the most probable meaning for each morph. The deterministic preprocessor does absolutely everything else to prepare for the probabilistic or stochastic part. There are two important pieces of input for the deterministic side namely the corpus (KeLaCo) and the morphological information needed to create a grammar (KeLaGram).

At this point there are some intriguing questions whose answers are to be found in the text. Section one takes a look at morphology and Latin morphology in particular as well as some basic statistical concepts. Although these two topics do not seem to go together, they comprise the basic knowledge needed to understand the project. The next two sections describe the major pieces of the hybrid system. One of the most essential parts is of course data to work on, so section two explains the construction and formatting of

the classical Latin corpus (KeLaCo). Section three describes how KeLaGram, the classical Latin Grammar developed for this project is created and used. These sections combine to describe the deterministic preprocessor and the probabilistic postprocessor. Section four gives a recap of the system. Section five reports on the two experiments used to test the feasibility. The final section gives an overview of future work. This includes plans for experiments beyond the scope of this project and general benefits of such a line of study.

I Background: Morphology for Statisticians and Statistics for Linguists

I.1 Morphology

Morphology is the study of the smallest units in a word which carry meaning. Languages exploit morphology to varying extents. We are here concerned with Inflectional Morphology, which is used like cement, to help provide a framework in which the relations between lexical items can be made clear. In other languages syntax (word order) performs many of the same functions. Hence the word morphosyntax which is used to refer to the sum of the functions of inflectional morphology and syntax[2]. In a language like English, syntax/word order does a lot of the work, while in a highly inflected language like classical Latin or modern Polish, morphology does the work, allowing word order to be freer. For instance, in English it is obligatory that the subject come before the verb and the object after. In latin, there is a specific morph marking the word which is the subject and that which is the object allowing them to come at any point in the utterance without confusion. Highly inflected languages are often more compact, with a most of the information found in the endings. As a simple comparison take the English sentence 'I walk' and the equivalent Latin sentence 'ambulo'. Here the information that a first person subject is involved is contained in the –o ending.

Latin is a Proto-Indo European language with a rich morphology. For verbs, suffixes denote person, number, mood, tense, aspect, voice, and occasionally case and gender (participles). Latin noun suffixes

[2] A thank you to Norval Smith who put that far more eloquently than I could have.

carry information regarding case[3], number, and gender. Thus it is easy to think of a Latin word in two parts. The stem or main part contains the lexical semantic information i.e. what the word means. The second suffix part contains all the grammatical information. This is the part we are concerned with here. Note well that we are working with a suffix only version of the corpus and no lexical semantic content was taken into account.

I.2 Statistics

For those of us with a clear sense of morphology but only a vague notion of statistics, here is a brief synopsis highlighting some important terms. The typical way for statistics and linguistics to interact is to start with a corpus. This corpus is generally large and fairly uniform[4] and painstakingly hand-annotated with respect to various linguistic information. A *probabilistic* grammar is then generated by reading the grammar rules directly off the corpus as instances of them appear. The probabilities of the grammar rules are found either by straight counting or relative-frequency estimation. In this sort of project, the corpus, often called a Treebank, is created by many people for the general benefit of anyone who wants to study what they have tagged. Tagging is synonymous with hand-annotating. Corpora can be tagged with respect to syntax, semantics, and discourse structure to name a few. Bascially any linguistic annotation one can think of and hand-annotate or tag a large corpus with can be done. This project would require part of speech (POS) tagging if it were available. This type of work is known as supervised (grammar) training and has been proven to be very useful in a number of areas. It is not however, the topic of this project.

The alternative to this sort of work is unsupervised training which starts with a grammar and the rule probabilities are learned from a corpus of ambiguous data (supervised training deals with unambiguous data, unambiguous because everything is tagged). Maximum-Likelihood Estimation then selects the instance of a probability distribution for the grammar that has the maximum likelihood of producing the original corpus[5]. This very important concept comes in a variety of flavors. In fact Maximum-Likelihood estimation has a different algorithm corresponding to each of the different grammar formalisms, and these

[3] Grammatical case refers to inflections (suffixes) which denote a word's relation to other words in the sentence. For example, the nominative case in Latin denotes the subject of the sentence while the accusative marks the direct object.

[4] Most corpora have large amounts of the same type of tokens like newspaper articles or recordings or hotel reservation bookings.

are generally bundled under the term Expectation-Maximization (EM). The EM algorithm was designed to replace relative-frequency estimation in places where there is not enough data to compute it. For finite state grammars it is realized as the Forward-Backward algorithm. And for Probabilistic Context Free Grammars (PCFG) it is the Inside-Outside algorithm (I-O).

All of the algorithms above estimate the probability for the individual rules of a grammar using annotated (relative-frequency) or un-annotated (EM) corpora. For our purposes, as you will see below, we must also find the most probable parse for an entire sentence given that sentence and a probabilistic grammar. The Viterbi algorithm does this for us. In both supervised and unsupervised learning it is important to note the underlying assumption that the corpus itself contains linguistic information. This falls in line with the notion that instead of a learning a set of grammar rules or (perhaps in combination with) a ranked series of constraints, speakers and hearers draw upon their past language experience when determining the meaning or use of an utterance.

Recent work in probabilistic context-free grammars (Schulte im Walde 2000, Mariani 1999) has opened up this formalism often thought to be exclusively associated with phrase structure. CFGs contain a series of rewrite rules that would at first glance look exactly like the phrase structure rules you are familiar with. One of the things explored in this project is to utilize a PCFG in developing a statistical morphological analyzer. This line of natural language processing (NLP) being relatively new, there are few choices in software. LoPar offers parsing of probabilistic context free grammars and head-lexicalized probabilistic context free grammars (HPCFGs) as well as Viterbi parsing in a *linguist*-friendly environment.

The power of the I-O algorithm was tested in a number of ways. In the following experiments, the sparsest of data was used. As there wasn't a sufficient dictionary/lexicon available for Latin, the suffix/morphological ending only was used in determining the part of speech and linguistic tags. As there is no existing tree bank for Latin, grammar training was unsupervised and a test corpus had to be hand-annotated for parser evaluation. The left-corner parser LoPar was used to instantiate the I-O algorithm and Viterbi parse. It touched upon more directly in section III.

---

[5] For more mathematical information on these concepts see Prescher 2003

II. KeLaCo: Corpus Preparation

II.1 Benchmarks and New Horizons

As mentioned in the introduction, linguistic investigation is traditionally done on large annotated corpora. As this type of resource was not available for Latin, a sufficient corpus needed to be created for unsupervised training. While there is no Treebank (annotated corpus), classicists have indeed found the web and so there is a vast amount of Latin material available. Although there are surely better ways, time constraints indicated it was best to do it by hand. Over the course of a week I prepared a corpus of 1.2 million words, hereafter KeLaCo. In this study only classical Latin texts were used including both prose and poetry. For a complete list of the authors included see Appendix A.

This is somewhat different from the corpora typically used in NLP studies. The Penn Treebank of over 1 million hand tagged words is the largest Treebank and generally considered the benchmark standard. Other large corpora include the Huge German Corpus (418,290 verb-noun or adj-noun pairs tagged) for German, for Italian the Turin University Treebank (TUT) (1,500 sentences), and the Prague Dependency Treebank (PDT)(456,705 tokens in 26,610 sentences) for Czech. All of these are mainly comprised of newspaper articles. Of course any corpus can be turned into a Treebank. With just a few steps we can see the Treebank version of such favorites as The Hansard Corpus consisting of parallel texts in English and Canadian French, drawn from official records of the proceedings of the Canadian Parliament or DCIEM the Sleep Deprivation Corpus (ready with transcription!).

These corpora while complex, are obviously fairly uniform, a benefit to their training, meaning that there are fewer different types of rules or tokens and more instances of them to make the probability calculations more precise. Another important feature, is that they have been painstakingly hand-annotated. I cannot emphasize enough the hard exacting work that goes into annotating a corpus.

This reveals two challenges our corpus has to offer. First, while one may object that poetry contains irregular syntax and unusual forms, I personally have an interest in an ability to handle this situation. Success would indicate that this technology could be useful to historical linguists who may not be interested in the finer points of probability theory. Although there may be some archaic forms to be found, we want to be able to handle this. Also the general free word order nature of Latin is such that poetic license is not all that threatening to our cause. Unlike other corpora there is not a single unifying theme to

the texts. Histories, personal letters, some poetry, treatises on nature and philosophy, are all included in KeLaCo.

The second noteworthy aspect of KeLaCo is that its quick compilation means that the corpus is not hand-tagged. In the experiments below, the test set was removed from the training data beforehand and personally hand-tagged. This is unforgiving work; indeed, I was only able to tag 50 of the 200 test sentences in the set. This tagged set included over 100 each of verbs and nouns. This means that while we can get a good indication of how our grammar has performed, finer distinctions as in a percent plus or minus are unattainable until more tagging can occur. While it is disappointing to have such sparse data to test on, the training environment can easily be seen as an advantage. Similar experiments could without difficulty be set up for Ancient Greek, possibly Sanskrit, for languages where there isn't the number of people with the expertise to hand-tag and the amount of material is not considerable (or ever going to get any bigger).

A less positive aspect of KeLaCo's preparation is that there are bound to be errors. There are always problems with transcription and I cannot guarantee that the base Latin was one hundred percent correct. That is to say that for each individual website various methods were used to digitize the text. Transcription by hand has been plagued by errors as long as it has existed. For an old language like Latin this means there have been opportunities for scribal error many many times.

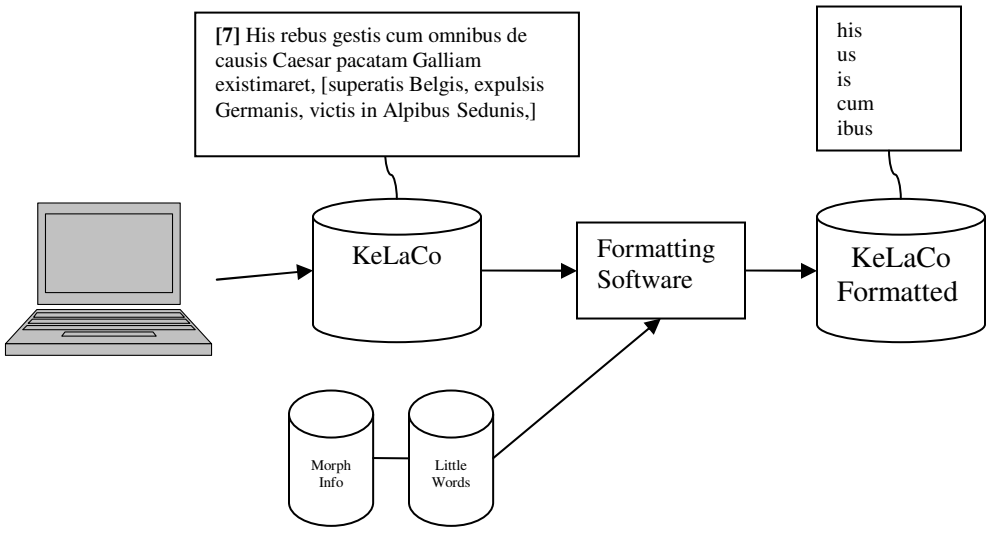


Figure 2 Corpus Preparation

II.2 Construction and Formatting

All of the texts were obtained on-line. Each had its own special formatting with the possibility of line numbers, lacunas, abbreviations, roman numerals, and comments. A number of Perl scripts were used to remove this extraneous material and put each word on a line. Blank lines were inserted after periods, question marks, and exclamation points to indicate the sentence boundary. See Appendix B for samples of the text before and after formatting. I chose the more agreed upon and complete texts available for the classical period. Therefore, lacunas were rare. There was also a script which removed enclitic que ('and') and replaced it with –que which was added to the little words file as a conjunction. Great difficulty arose with enclitic ne (indicates a question) because it occurs in different environments than que and is even part of some morphs[6].

II.3 Why Suffixes?

While I needed the full words to hand-annotate the test set, the learning data needed to be formatted slightly differently. The basic format is still one word per line with a blank line indicating sentence boundaries. However, if you recall the second module aims to reduce ambiguity among the different meanings for each Latin morph. For this reason, the lexicon of KeLaGram is made up of morphs, and so the training data must contain the information-rich endings only. To accomplish this, the deterministic preprocessor (another Perl script) printed out the endings only into a separate file used as input for the LoPar experiments. At this point you may have some doubts about the challenges we are putting forward. Not only is the corpus untagged and uncertain, it doesn't even have whole words. However after a moment's consideration this can be seen as an advantage, perhaps another challenge. For our analysis, the complete word could only complicate matters. There isn't a sufficient (available) Latin dictionary which could be added to the machinery of the parser for disambiguation. As discussed in section three, LoPar requires a lexicon for parsing. Instead of compiling a dictionary's worth of information, our lexicon contains only the morphological endings found in Latin. It is in this way that LoPar can emulate a morphological analyzer at all. It is literally the difference between hundreds of thousands of entries and a few hundred.

[6] This would indicate that words with enclitic ne are not correctly analyzed.

II.4 How the Suffixer Works

As part of the deterministic preprocessor, the suffixer (yet another Perl program) isolates the meaningful morph ending. This section describes the creation and function of the suffixer. By hand two large files containing the important information were developed. The first of these files contains the complete morphology for nouns of all five declensions, adjectives from the three declensions along with third declension I-stems (the tags for nouns and adjectives of the first and second declension have been collapsed to NA since the endings completely coincide and adjectives can be substantive). Verbs in all four conjugations have been accounted for in all forms that do not take a form of the verb esse 'to be'. This includes limited participle forms as well as the infinitive. I have gone beyond what traditional classicists would call the root and ending portions of the verb to include the vowel alternations in verbs of differing declensions. This is of course the mysterious file 'Morph Info'.

The second file contains what those at the SGLI[7] would call little words. These are monosyllabic pronouns (reflexive, relative, interrogative, and regular), indeclinable adverbs (although some adverbs can be formed through a morphological change, those are beyond the scope of this project), conjunctions, and prepositions. This file also includes very irregular words like the verb 'to be' esse, 'to be able' possum, 'to go' ire, and the nouns for strength, home, and gods.

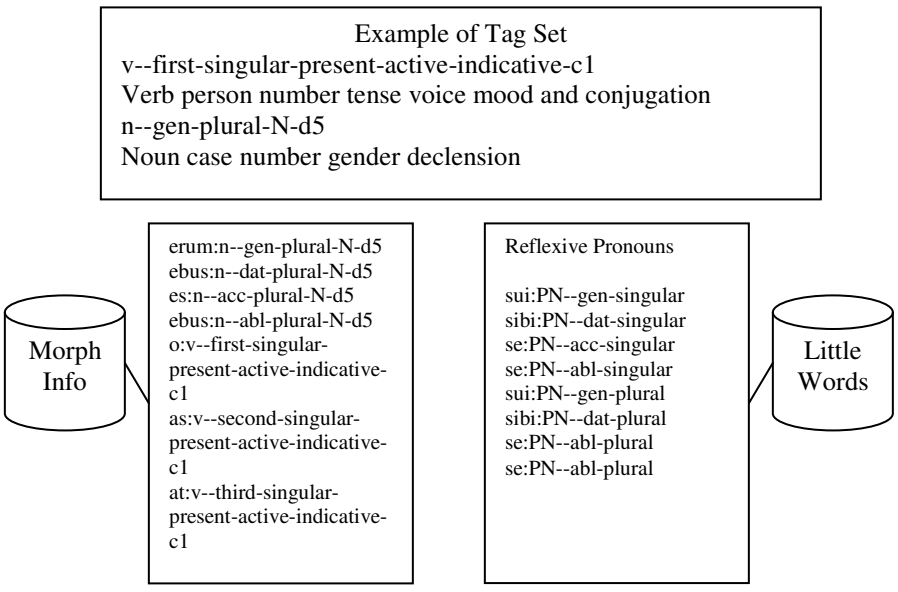


Figure 3 'Morph Info' and 'Little Words'

[7] Summer Greek and Latin Institute at CUNY

These files are the backbone of the whole project and so were checked thoroughly. Then began a series of Perl programs which would allow one to analyze a text. The shape of the data structure used by the analyzer was very important as there are multiple entries for each token morph. The best solution was to read in each line of the file with the morph as a hash key referencing an array of the different tags associated with the morph. Each file had a different hash for checking ease.

The suffixer actually works by reading in a line of the text, one word at a time. It then checks to see if that word is a 'little word'. If so, the entire word is added to the formatted, suffix-only KeLaCo text along with a newline, and the analyzer moves on. If not, one letter is removed from the front of the word and then the remainder is checked against the morph keys. This continues until a morph is found and in this case the morph is added to the text. The thought behind this is to find the longest possible ending and stop with it. In some cases this will produce incorrect results; however, in the portion hand checked I could not find any errors that weren't associated with enclitic ne. Great pains have been taken to remove extraneous information; nevertheless, some may have slipped by. The suffixer outputs anything it cannot understand into the suffix-only version of KeLaCo. The preprocessor also uses the 'Morph Info' and 'Little Word' files to provide a grammar and lexicon for the probabilistic portion of the processing as seen below.

III. KeLaGram:Probabilistic Grammar Preparation

Already there have been a number of jobs done by the deterministic preprocessor. These have been presented in the context of corpus preparation. This section will illuminate the rest of the deterministic preprocessor's role in the context of its second major task: to create KeLaGram.

Given a few adjustments, the suffixer described in the previous section could easily produce all the possible tags for a given morph, the basic task of a morphological analyzer. This would work very well; however, as each morph has a series of possible tags the output would include a wealth of ambiguity. In a course on Data Oriented Processing I was introduced to the EM algorithm and the possibility of adding statistics to linguistics. It was here that the idea began to form that these types of methods could be brought to bear on my ambiguity problem. In particular unsupervised training of a probabilistic grammar may be able to reduce ambiguity, and the Viterbi algorithm could force a best possible outcome for whole

sentences. Relative-Frequency estimation was an attractive option; however, KeLaCo is not annotated and therefore ineligible for relative-frequency estimation. There are many grammar formalisms to choose from each with its own parsers and estimation algorithms. For a variety of reasons LoPar seemed the best tool for the job. LoPar is a parser which uses the Inside-Outside algorithm to compute maximum-likelihood on PCFGs. This in mind, it became my task to transform the information utilized by the suffixer into a context-free grammar.

The deterministic preprocessor has the power to isolate and assign all the possible tags to a given morph. It does not however have any way to distinguish between all those possible tags. This is where LoPar and the Inside-Outside algorithm come into play. By working within this framework, I could take advantage of the statistical methods widely used today in a tested environment.

In fact, using this particular formalism will allow me to cut down on ambiguity in two ways. First, the whole program is meant to learn the correct probabilities and then give all possible or simply the most likely parse. This normally wouldn't be involved in morphological analysis; however, it is a fact that the ending ibus which can denote third declension plural dative or ablative case is far more often ablative, and LoPar can learn that. Secondly, as I had originally wanted to resolve some ambiguity LoPar gives me the ability to use some small bits of syntax to that end. This is because of the setup of the grammar file explained below.

The LoPar parser requires a number of files which will be enumerated then discussed more fully. These include two open class files (one for upper case one for lower case), a lexicon, a grammar file, a start file, and a corpus. All of these files are bundled as KeLaGram. Although there is a grammar file described below, it doe not form the full KeLaGram classical Latin grammar without the lexicon and other files.

Since this formalism is being adapted to the task of morphological analysis, this particular grammar and the lexicon are a bit unique. Again the original files containing the morph information and the little and\or irregular words were used as the starting point. The lexicon was made using LexBuilder (part of the deterministic preprocessor) which built one large data structure (thereby removing morphs that were duplicate because they were both an ending and a little word) and outputting it to a LoPar friendly lexicon with each tag given an arbitrary frequency of 100. It also added the pertinent punctuation with a frequency of 1000. GramBuilder (also part of the preprocessor) constructed the grammar file in a similar

way, except that the tag and not the morph was outputted with “DUMMY_SYNTAX” on either side. (see figure 4) Rules for the top node and ending punctuation were also included to begin and end the fake parse of the sentence. Each rule was given a starting probability of 1. The deterministic preprocessor also ran grammar file through a small script to remove duplicate rules.

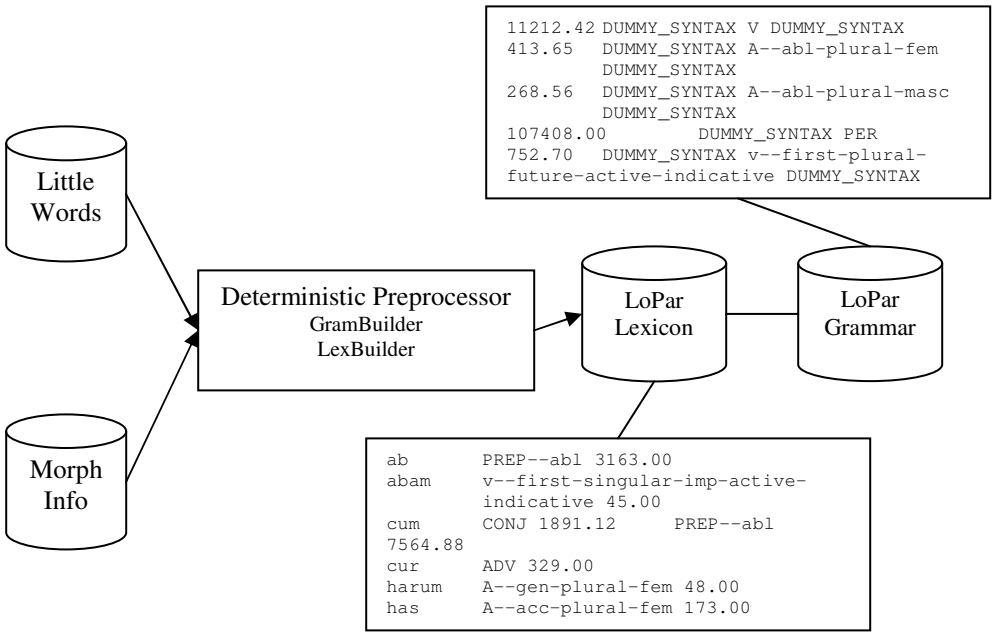


Figure 4 LoPar Lexicon and Grammar the samples here are taken from a trained version of the grammar and lexicon for experiment 2.

The start file contains only those symbols which are acceptable as root nodes in a parse tree. For this experiment using dummy syntax, there is only one entry:

TOP 1000

As discussed above there are inevitably certain words which will have escaped the suffixer. In order for LoPar to handle unknown words, the open class file contains word categories which unknown words are likely to be. So, a LoPar lexicon is complete with respect to words like conjunctions and prepositions but not necessarily nouns. The lexicon used in this project is closed with respect to prepositions, conjunctions, and pronouns. It contains all of the information outlined above. Most notably, open class words will be errors in formatting or nominative noun forms which may not have an ending.

KeLaCo is formatted to be lower case so only one open class file is required (the other must be present but is left blank). As an example, the following was used as the first test in open class files:

N 1000

V 10

ADV 100

These five files comprise KeLaGram, the Latin PCFG for use with the probabilistic postprocessor in resolving ambiguity.

IV. Hybrid System

Now that all the pieces are in place, it may be appropriate to revisit Figure 1 and see what we have accomplished.

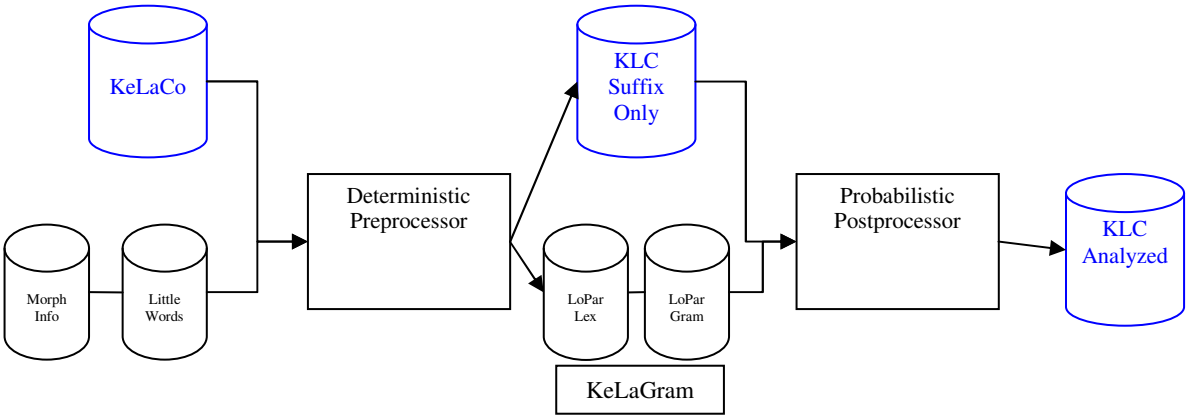


Figure 1 The Big Picture revisited

So, the majority of the work came in the first half of the picture in preparing a large corpus KeLaCo and the files containing the crucial Latin morphological paradigms. The large behind the scenes work came in writing the Perl programs which I have put under the heading Deterministic Preprocessor. The hybrid element of the whole thing is that I use both deterministic and probabilistic tools to bear on the problem. Probability is used only to disambiguate among the choices isolated by the deterministic preprocessor and realized for the postprocessor in the lexicon portion of KeLaGram. This probabilistic postprocessor takes in KeLaGram as well as the prepared suffix-only version of KeLaCo and returns the corpus with each word given its most probable tag. It does this using the Inside-Outside algorithm touched upon in the first section to determine the probability for each rule in the context-free grammar making it a

probabilistic context-free grammar. The Viterbi algorithm then finished the job by picking the most probable for each word/morph in each sentence.

This is a somewhat novel way to approach the problem. Its advantages are varied. The opportunity to study what a corpus has to offer and how I can apply that knowledge to my morphological analysis is not least on the list. Another is testing the boundaries of the PCFG, using different techniques to further refine my findings. The following section describes two experiments conducted to test the feasibility of the system. The final section offers some directions for further work.

V. Feasibility Study: 2 Experiments

With KeLaCo and KeLaGram in hand, the system is ready to be tested. As stated above, LoPar uses the Left-Corner Algorithm for parsing and the Inside-Outside algorithm for parameter estimation. After training, a probabilistic version of the context-free grammar is obtained and ready for evaluation. This evaluation is twofold. There is both the mathematical evaluation in the measure of perplexity as well as the linguistic evaluation: did it judge the morph correctly.

Perplexity is a measure of fitness for a model. Formally, the perplexity of a corpus C with respect to the model of the language M is captured in the following formula:

$$Perp_M(C) = e * -\log P_M(C) / N$$

Here N is the size of the corpus (i.e. number of tokens), and $P_M(C)$ is the likelihood of the corpus according to the model. In layman's terms, the perplexity of the model is a number corresponding to the difficulty in choosing the correct ending of the next word. So, if the perplexity is 72 then there is as much difficulty as when choosing among 72 equal candidates. Clearly the perplexity should be the highest at the beginning and decrease during training

The linguistic evaluation of these experiments is of course whether or not the morphs were correctly analyzed. While LoPar is certainly capable of handling more complex tasks such as German noun chunking (Schulte im Walde 2000), our goal is to see if the CFG formalism can be used to the advantage of a morphological parser.

In the first experiment, the grammar as taken directly from the 'morphs' and 'little words' files is used to a not unsatisfactory result. In the second experiment, the grammar was reduced to eliminate some possibly superfluous information and perhaps increase the likelihood of effective learning.

V.1 Experiment One

A. Training Environment

The training environment has been determined by all of the corpus preparation and analyzer transformations described above. LoPar requires file names with the same prefix and the following suffixes corresponding to the files discussed above.: .start, .lex, .gram, and .oc/.OC.

B. Training Strategy

The training text was composed of 90% of the sentences in the KeLaCo corpus. The testing corpus contained the remainder, and they were named klctrainer.txt and klctester.txt respectively. The training was conducted as follows:

i) Initialization:

All context-free grammar rules were initialized with the identical frequency of 1.

ii) Unsupervised Training

The training corpus was parsed by Lopar once in five iterations with the probabilities re-estimated each time.

iii) Parsing of the Test Corpus

After checking for optimal training (where there is no change in the output perplexity and actual parse as well as no over-training), the test corpus is parsed with the probabilistic grammar.

iv) Evaluation

After parsing is complete results are compared to the hand annotated file using the Evaluation Suite (part of the probabilistic postprocessor) developed specifically for this purpose.

C. Probability Model Evaluation

For linguistic evaluation of this experiment, the test corpus was hand annotated. This was the reason that the training corpus and the test corpus went through the suffixer separately. Otherwise it would be very difficult to obtain the original sentences for hand annotation. The evaluation software first extracts the

pertinent information from the LoPar output. As its nature is a syntactic parser, its output consists of ‘trees’ which actually look like this:

```
(TOP (DUMMY_SYNTAX (PN--nom-singular-fem quae)(DUMMY_SYNTAX (PREP--abl
cum)(DUMMY_SYNTAX (na--nom-singular-M-d2 us)(DUMMY_SYNTAX (v--second-singular-present-
active-indicative-c4 is)(DUMMY_SYNTAX (v--second-singular-present-active-indicative-c4
is)(DUMMY_SYNTAX (PREP--acc per)(DUMMY_SYNTAX (na--acc-plural-M-d2 os)(DUMMY_SYNTAX
(v--second-singular-present-active-indicative-c4 is)(DUMMY_SYNTAX (v--third-singular-
pluperfect-active-subjunctive-c3 isset)(DUMMY_SYNTAX (CONJ ut)(DUMMY_SYNTAX (v--
second-singular-perfect-active-subjunctive-c4 eris)(DUMMY_SYNTAX (na--dat-singular-F-
d1 ae)(DUMMY_SYNTAX (na--nom-singular-N-d2 um)(DUMMY_SYNTAX (v--third-singular-imp-
passive-subjunctive-c1 aretur)(DUMMY_SYNTAX (PN--abl-singular-masc quo)(DUMMY_SYNTAX
(na--nom-singular-M-d2 us)(DUMMY_SYNTAX (na--nom-singular-N-d2 um)(DUMMY_SYNTAX (n--
gen-plural-N-d5 erum)(DUMMY_SYNTAX (CONJ -que)(DUMMY_SYNTAX (part--present-active-nom-
singular-N ns)(DUMMY_SYNTAX (v--imperative-singular-present-active-c1 a)(DUMMY_SYNTAX
(v--imperative-singular-present-active-c3 e)(DUMMY_SYNTAX (PREP--acc ad)(DUMMY_SYNTAX
(v--first-singular-future-active-indicative-c4 iam)(DUMMY_SYNTAX (v--third-singular-
imp-passive-subjunctive-c2 eretur)(DUMMY_SYNTAX (v--third-singular-present-active-
indicative-c2 et)(DUMMY_SYNTAX (PREP--abl ex)(DUMMY_SYNTAX (n--dat-plural-N-d4
ibus)(DUMMY_SYNTAX (n--gen-singular-F-d3 is)(DUMMY_SYNTAX (n--acc-singular-M-d4
um)(DUMMY_SYNTAX (PREP--abl ex)(DUMMY_SYNTAX (n--acc-singular-M-d4 um)(DUMMY_SYNTAX
(CONJ autem)(DUMMY_SYNTAX (v--first-singular-present-active-subjunctive-c1
em)(DUMMY_SYNTAX (n--abl-singular-M-d3 e)(DUMMY_SYNTAX (na--dat-singular-F-d1
ae)(DUMMY_SYNTAX (PREP--abl de)(DUMMY_SYNTAX (v--imperative-singular-present-active-c1
a)(DUMMY_SYNTAX (v--imperative-singular-present-active-c1 a)(DUMMY_SYNTAX (n--gen-
plural-M-d3 ium)(DUMMY_SYNTAX (v--third-plural-imp-active-subjunctive-c2
erent)(DUMMY_SYNTAX (v--second-singular-present-active-indicative-c4 is)(DUMMY_SYNTAX
(v--imperative-plural-present-active-c1 ate)(DUMMY_SYNTAX (v--imperative-singular-
present-active-c1 a)(DUMMY_SYNTAX (v--third-singular-present-active-indicative
est)(DUMMY_SYNTAX (PER .))))))))))))))))))))))))))))))))))))))))))))))))))
```

Lopar Output for One Sentence

After extraction the output looks like this:

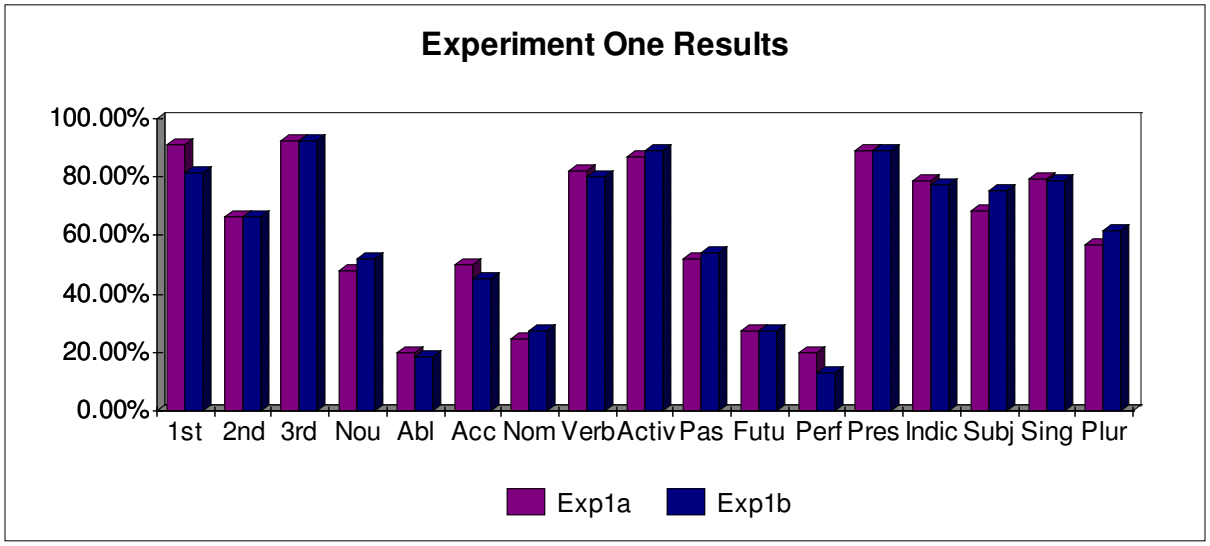
```
ut      CONJ
eris    v--second-singular-perfect-active-subjunctive-c4
ae      na--dat-singular-F-d1
um      na--nom-singular-N-d2
aretur  v--third-singular-imp-passive-subjunctive-c1
quo     PN--abl-singular-masc
us      na--nom-singular-M-d2
um      na--nom-singular-N-d2
erum    n--gen-plural-N-d5
CONJ -que
ns      part--present-active-nom-singular-N
a       v--imperative-singular-present-active-c1
e       v--imperative-singular-present-active-c3
ad      PREP--acc
iam     v--first-singular-future-active-indicative-c4
eretur  v--third-singular-imp-passive-subjunctive-c2
et      v--third-singular-present-active-indicative-c2
ex      PREP--abl
ibus    n--dat-plural-N-d4
is      n--gen-singular-F-d3
um      n--acc-singular-M-d4
ex      PREP--abl
um      n--acc-singular-M-d4
autem   CONJ
em      v--first-singular-present-active-subjunctive-c1
e       n--abl-singular-M-d3
ae      na--dat-singular-F-d1
de      PREP--abl
a       v--imperative-singular-present-active-c1
a       v--imperative-singular-present-active-c1
ium     n--gen-plural-M-d3
erent   v--third-plural-imp-active-subjunctive-c2
is      v--second-singular-present-active-indicative-c4
ate     v--imperative-plural-present-active-c1
a       v--imperative-singular-present-active-c1
est     v--third-singular-present-active-indicative
.       PER
          Same Sentence After Formatting
```

And it is ready for a linguistic evaluation. Various tests can be performed and the method is simple. Both the LoPar output and the hand annotated file are put though a file which marks a yes or a no next to each form based on its tag. As these files are mirror images, they can be pasted together to form a master test file. The evaluator then checks to see if the hand annotated file has marked a form with YES and if the LoPar output has matched that evaluation. The evaluator gives statistics of how many LoPar got correct out of the number possible. In this way it is simple to merely change the parameters and check for anything that had been tagged.

```
quae    PN--nom-singular-fem   NO      quae    NO
cum     PREP--abl      NO      cum     CONJ    NO
us      na--nom-singular-M-d2  NO      senatus N masc nom sg  NO
is      v--second-singular-present-active-indicative-c4      YES     libris          NO
is      v--second-singular-present-active-indicative-c4      YES     sibyllinis      NO
per     PREP--acc      NO      per     PREP acc       NO
os      na--acc-plural-M-d2    NO      decemuiros     N masc acc pl  NO
is      v--second-singular-present-active-indicative-c4      YES     inspectis       V
PART part pass masc abl pl    YES
isset   v--third-singular-pluperfect-active-subjunctive-c3   YES     censuisset      V
3rd sg plup subj act   YES
ut      CONJ    NO      ut      CONJ    NO
eris    v--second-singular-perfect-active-subjunctive-c4     YES     veneris V 2nd sg
futperf ind act        YES
ae      na--dat-singular-F-d1  NO      verticordiae   NO
um      na--nom-singular-N-d2  NO      simulacrum     N neut nom pl  NO
aretur  v--third-singular-imp-passive-subjunctive-c1 YES     consecraretur  V 3rd sg
imperf subj pass       YES
quo     PN--abl-singular-masc  NO      quo     PN abl sg      NO
us      na--nom-singular-M-d2  NO      facilius               NO
um      na--nom-singular-N-d2  NO      uirginum       N fem gen pl   NO
erum    n--gen-plural-N-d5     NO      mulierum       N fem gen pl   NO
CONJ -que      NO      -que    CONJ    NO
```

Example of a Master Tag Set (this one marking verbs)

Actual Data:



In the first experiment, the grammar of 731 rules was trained and re-estimated 20 times. The perplexity after the first iteration was 516.028. After the second iteration it dropped to 92.6171 and stayed there through the 20th iteration. There was linguistic variation to be noticed though which warranted continued training. By the end it was clear that there was only a single linguistic change in the Viterbi parse of the sentence. This was due to two rules having only the slightest difference in probability. The output of both the 19th and 20th grammars is shown in the chart above as Exp1a and Exp1b respectively.

The chart above gives the results for all the categories tested. It is important to remember that the test corpus is very small, so in some cases there were only a few tokens to judge against. Still, this is encouraging for the first try. 'Little Word' categories such as pronoun were not tested for as they cannot have been analyzed incorrectly. While verbs were found correctly over 80% of the time, nouns were found less than 60% of the time. Within the verb category, mood had good results at over 80%; voice was in the middle, active was found over 80% of the time but passive (a more marked form) was found only 52% of the time. Tense scored 3/11 future and 65/73 present.

V.2 Experiment Two

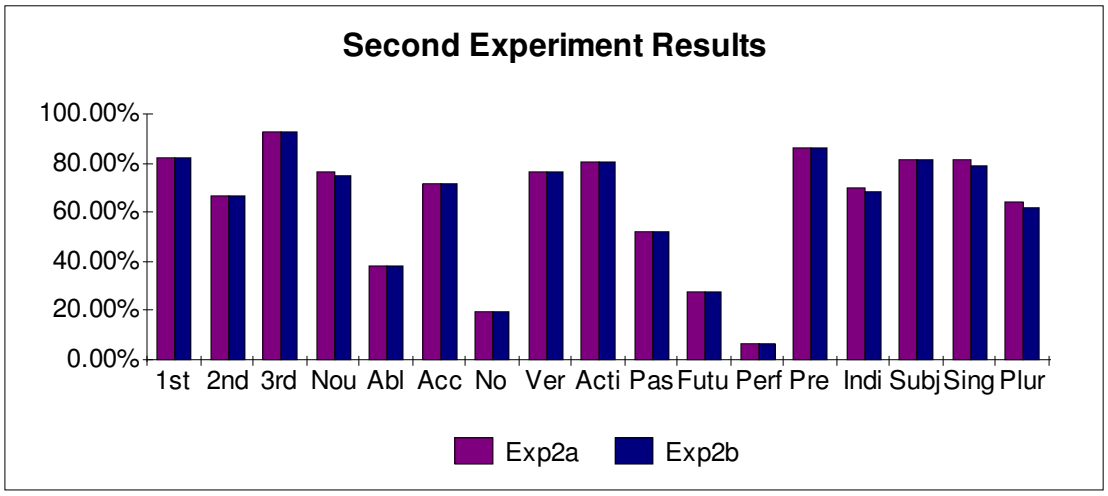
A. Training Environment

The training environment was identical to that in experiment two in all but one variable. The 'Morph Info' file was altered to remove information about declension and case. Many rules were duplicate for just this reason (5 declensions, 4 cases). The number of rules in this grammar is only 306 rules compared to the 731 above (neither is a very large figure there are over 5000 rules in the phrase structure grammar of German)

B. Training Strategy

Again the training strategy is as above and does not require reiteration.

C. Probability Model Evaluation

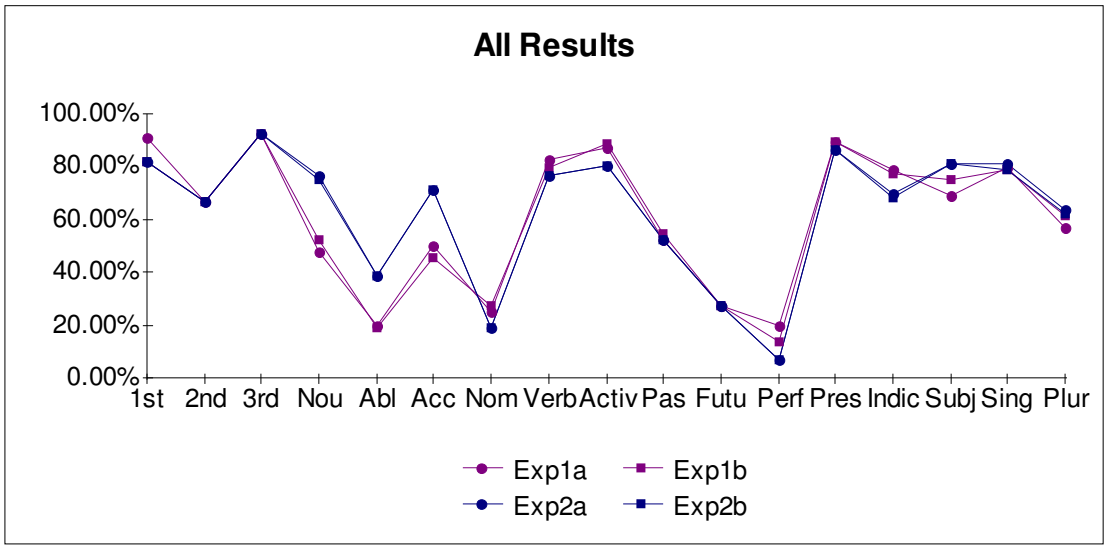


In the second experiment, the analyzer found 102 out of 136 nouns and 83 out of 109 verbs. In those categories it correctly marked 27/70 ablative forms, 30/42 accusative forms, and 14 of 73 nominative forms. It found 63 of 68 3rd person verbs, 2/3 second person and 9 out of 11 third person verbs. It found 2

of 30 perfect tense verbs and 63 of 73 present tense. Singular forms varied in the two trials by 3% or 15 vs. 152 out of 193. Plural forms were found about 62% of the time.

V.3 Discussion

These two experiments were intended to test the feasibility of this system. Below are the results of the experiments as a review of that above and the basis for discussion.



In the first experiment, the results were very encouraging. Some categories had low scores as was to be expected. For example nominative noun forms often do not have an ending, so it was not surprising to find that only 18-20 out of 73 had been correctly identified. Also there were only 3 second person verb tokens to be found in the test set, so a 66% really means that only one was incorrectly identified. In any case this is too small a number to base any scientific finding on. It does however indicate that this is a feasible course of study. In general the scores for noun and case were very low. Only 65 – 71 out of 136 nouns or about 50% were correctly identified as nouns. The very common ablative case had the lowest score within nouns at only 20%. Still in this first experiment many forms were correctly identified suggesting grammar improvement could be worthwhile.

The results of experiment two show that grammar reduction did improve some scores. Most notably, noun identification rose by 25%. All cases showed improvement with accusatives being identified about 70% of the time up from 50%. In both experiments perfect forms were almost never correctly identified. Yet the

other tenses often were. As I wasn't completely sure what to expect, the results of these experiments are very interesting. Future work in improving the grammar could shed light on why and how some things were learned better than others. It is also important to remember that the testing corpus was very limited. Judging performance on only 3 or 11 tokens is not enough. These results are encouraging though and prompt both grammar improvement and more hand-tagging.

VI. Future Work

In the two feasibility trials above, a PCFG was used but barely exploited. The grammar produced by the deterministic preprocessor for these experiments was what is known as a unigram model and contained no nontrivial rules. It came in this form:

DUMMY_SYNTAX n—nom-singular-masc DUMMY_SYNTAX

There are a number of ways to introduce meaningful rules into the grammar and thereby reduce ambiguity.

A very successful model in the statistical NLP world is the n-gram model. These models are based on estimates of strings of words so that the probability of a word given the preceding word or words is approximated to the probability given the preceding words. KeLaGram as used in the experiments above is a unigram model, it does not take other words into account. The bigram model which takes into account one preceding word is the most successful n-gram model. The deterministic preprocessor could be altered to give the appropriate grammar rules with relative ease. The system needs no other altering, anything done from now on is grammar improvement which translates to a Perl script which manipulates the morph data files into the desired grammar. Due to the free word order and some instinctive notions about Latin, it may be worthwhile to test a trigram model. This is generally restricted to uniform corpora because it involves dividing the number of times a triplet of words is found by the number of times the pair beginning that triplet is found. There is rarely enough data to form accurate estimates.

After testing out the bigram models, adding syntactic information to the grammar could prove very useful. For instance, prepositions (all of which are accounted for) take a certain case. This sort of rule is fairly simple especially working off the grammar of the second experiment which does not contain declension information. The deterministic preprocessor could be altered to print out a series of rules that boil down to a preposition only being able to take ablative or accusative in the next word.

It is often noted that CFGs and free word order languages do not mix well. There has, however, been some work done to capture scrambling in a HPCFG (Schulte im Walde 2000) which may be useful to our cause. A similar method could be employed to catch Latin noun phrase heads which are not required to be in a certain position but often line up in a traditional SOV way. This tactic seems particularly promising and development is underway. Finally, it would seem silly not to exploit the PCFG's most noted abilities as a phrase structure grammar. Adding syntax and particularly recursive syntax to the grammar would help to reach its full potential.

APPENDIX A Corpus Selection

KeLaCo contains internet versions of the following texts:

Apuleius
Cupid and Psyche [Libellus]

Ausonius
Mosella [Libellus]

Gaius Valerius Flaccus Sentinus Balbus
Argonautica [Forum Romanum]

Caesar
Bellum Gallicum [Libellus]

M. Tulli Ciceronis
Laelius De Amicitia[8] [Bibliotheca Augustana]
De Inventione [Bibliotheca Augustana]
De Ratione Dicendi ad C. Herennium [Bibliotheca Augustana]

C. Sallustii Crispi
Invectiva in Ciceronem [Bibliotheca Augustana]

Aules Persius Flaccus
Saturarum liber [Bibliotheca Augustana]

Grattius Faliscus
Cynegetica[9] [The Latin Library]

Julius Caesar Germanicus
Aratea[10] [The Latin Library]

Horatii
Epodon Liber [Bibliotheca Augustana]
Epistulae [Bibliotheca Augustana]

Gaius Julius Hyginus
De astronomia [The Latin Library]
[Pseudo-] Hygini De Munitionibus Castrorum [The Latin Library]
Fabulae [The Latin Library]

Livius
Ab Urbe Condita, author's preface [Libellus]
Ab Urbe Condita I [Libellus]

Marcus Annaeus Lucanus

---

[8] This version of the De Amicitia is a conflation of two texts: M. Tulli Ciceronis Laelius De Amicitia, ed. Clifton Price, 1902; and M. Tulli Ciceronis Scripta Quae Manserunt Omnia, Part 4, Vol. 3, ed. C.F.W. Mueller, 1890.
[9] ed. P.J. Enk (1918)
[10] ed. A. Breysig (1899)

De bello civili sive Pharsalia [The Latin Library]

Titus Lucretius Carus
De rerum natura [The Latin Library]

Lygdamus
Elegiae[11] [Bibliotheca Augustana]

Marcus Valerius Martialis
Apophoreta [The Latin Library]
Epigrammata[12] [Bibliotheca Augustana]
Liber de spectaculis[13] [Bibliotheca Augustana]
Xenia[14] [Bibliotheca Augustana]

Valerius Maximus
Facta et dicta memorabilia[15] [The Latin Library]

Nemesianus
L. Iunius Moderatus Columella [The Latin Library]
De Re Rustica Libri XII [The Latin Library]
De Arboribus [The Latin Library]

Nepos
Preface [Libellus]
Agesilaus [Libellus]
Aristides [Libellus]
Cato [Libellus]
Chabrias [Libellus]
Cimon [Libellus]
Conon [Libellus]
Dion [Libellus]
Hamilcar [Libellus]
Hannibal [Libellus]
Iphicrates [Libellus]
Lysander [Libellus]
Miltiades [Libellus]
Pausanias [Libellus]
Themistocles [Libellus]
Thrasybulus [Libellus]

P. Ovidi Nasonis
Amores [Bibliotheca Augustana]
Medicamina Faciei Femineae [Bibliotheca Augustana]
Ars Amatoria [Bibliotheca Augustana]

Phaedrus
Fabulae [Bibliotheca Augustana]

Seneca the Younger
L. Annaei Senecae Ad Neronem Caesarem De Clementia

---

[11] ed. Ulrich Harsch, 2001
[12] ed. Ulrich Harsch, 2000
[13] ed. Ulrich Harsch, 2000
[14] ed. Ulrich Harsch, 2000
[15] ed. Carolus Kempf (Leipzig: Teubner, 1888)

De clementia ed. Hosius[16] [The Latin Library]
De providentia [The Latin Library]
De constantia sapientis [The Latin Library]
De consolatione ad Marciam [The Latin Library]
De ira [The Latin Library]
De vita beata ad Gallionem [The Latin Library]
De otio [The Latin Library]
De tranquillitate animi ad Serenum [The Latin Library]
De brevitate vitae ad Paulinum[17] [The Latin Library]
De consolatione ad Polybium[18] [The Latin Library]
De consolatione ad Helviam matrem [The Latin Library]
Naturales quaestiones [The Latin Library]

Sulpicia
Epistulae [Bibliotheca Augustana]

Sallustius
Iugurtha [Libellus]
Catiline [Libellus]

Suetonius
The Lives of the Twelve Caesars[19] [LacusCurtius]
DeGrammatics [LacusCurtius]

Publilius Syrus
Sententiae[20] [The Latin Library]

Cornelius Tacitus
Agricola [ForumRomanum][21]
Annales ab excessu divi Augusti[22] [Latin Library]
Dialogus de oratoribus[23] Germania [Bibliotheca Augustana]

Albius Tibullus
Elegiarum liber primus (ca. 26) [Bibliotheca Augustana]
Elegiarum liber secundus (ca. 19) [Bibliotheca Augustana]

corpus Tibullianum:
Lygdami Elegiae [Bibliotheca Augustana]
Panegyricus Messallae [Bibliotheca Augustana]
De Sulpicia Incerti Auctoris Elegiae [Bibliotheca Augustana]
Sulpiciae Elegidia [Bibliotheca Augustana]
Incerti Auctoris Elegia [Bibliotheca Augustana]
Incerti Auctoris Epigramma [Bibliotheca Augustana]

P. Vergilii Maronis

---

[16] Leipzig 1900
[17] ed. Hermann Adolf Koch (Ulm: J.C. Wohler, 1884)
[18] ed. E. Hermes (Leipzig: Teubner, 1905)
[19] The Latin text is that of Maximilian Ihm in the Teubner edition of 1907, with cosmetic changes as printed in the Loeb Classical Library edition, 1913-1914. The English translation is by J. C. Rolfe, printed in the same edition. Both text and translation are in the public domain.
[20] ed. Eduard Woelfflin, Publilii Syri Sententiae (Leipzig: Teubner, 1869)
[21] ed. William Allen (Boston: Ginn & Co., 1913)
[22] ed. C.D. Fisher, Cornelii Taciti Annalium (Oxford 1906)
[23] ed. Henry Furneaux (Oxford: Clarendon Press, 1900)

Labores Iuveniles [Bibliotheca Augustana]
Eclogae [Bibliotheca Augustana]
Georgica [Bibliotheca Augustana]
Aeneis [Bibliotheca Augustana]

Vergiliana[24]
Elegiae in Maecenatem [Bibliotheca Augustana]

Velleii Paterculi
Historia Romana[25] [Bibliotheca Augustana]

Vitruvii
De Architectura [Bibliotheca Augustana]

These texts have been taken from the following sites:
[Bibliotheca Augustana] www.fh-augsburg.de(check this)
[Libellus] www.hhhh.org/perseant/libellus/texts/
[LacusCurtius] www.ukans.edu/history/index/europe/ancient_rome/E/Roman/Texts/
[ForumRomanum] www.forumromanum.org
Latin Library

[24] 'Haec carmina scripsit poeta ignotus saeculo primo ante Christum natum' www.augsberg.de
[25] book one only

APPENDIX B KeLaCo Formatting

3 sections of KeLaCo completely unformatted:

[7] His rebus gestis cum omnibus de causis Caesar pacatam Galliam existimaret, [superatis Belgis, expulsis Germanis, victis in Alpibus Sedunis,] atque ita inita hieme in Illyricum profectus esset, quod eas quoque nationes adire et regiones cognoseere volebat, subitum bellum in Gallia eoortum est. [2] Eius belli haec fuit causa. P. Crassus adulescens eum legione VII. proximus mare Oeeanum in Andibus hiemabat. [3] Is, quod in his loeis inopia frumenti erat, praefectos tribunosque militum eomplures in finitimas civitates frumenti causa dimisit; [4] quo in numero est T. Terrasidius missus in Esuvios, M. Trebius Gallus in Coriosolites, Q. Velanius eum T. Silio in Venetos.

IX

Quid mihi si fueras miseros laesurus amores,
Foedera per divos, clam violanda, dabas?
A miser, et siquis primo periuria celat,
Sera tamen tacitis Poena venit pedibus.

5

capitulum LI

ne prouincialibus quidem matrimoniis abstinuisse uel hoc disticho [FPR p. 330] apparet iactato aeque a militibus per Gallicum triumphum:

urbani, seruate uxores: moechum caluom adducimus.

aurum in Gallia effutuisti, hic sumpsisti mutuum.

A Section of KeLaCo test file formatted for hand annotation (same as LoPar with somewhat more informal tags):

```
harenas N fem acc pl
atque   CONJ
habilis ADJ fem nom sg
natura  N fem nom sq
soli
quae    PN fem nom sg
gramine N neut abl sg
laeto   ADJ neut abl sg
parturit        V 3rd sg pres ind act
et      CONJ
rutilas ADJ fem acc pl
ebuli   N neut gen sg
creat   V 3rd sg pres ind act
uvida   ADJ
bacas   N fem acc pl
.
```

N.B. LoPar is working on a suffix only version of the text.

BIBLIOGRAPHY

Bod, Rens Remko Scha Khalil Sima'an eds. (2003).*Data-Oriented Parsing* CSLI Studies in Computational Linguistics CSLI Publications.

Beesley, Kenneth R. and Lauri Karttunen. (2003). *Finite State Morphology*. CSLI Publications.

Jurafsky, Daniel and James H. Martin. (2000). Speech and Language Processing. Prentice Hall, Upper Saddle River, New Jersey.

Lari, K and S.J. Young. (1990) The Estimation of Stochastic Context-Free Grammars using the Inside-Outside Algorithm. *Computer Speech and Language*, 4:35-56.

Mariani, Joseph, Minker, Wolfgang and Alex Waibel. (1999) *Stochastically-Based Semantic Analysis*. Boston: Kluwer Academic Publishers.

Oakes, Michael P. (1998). *Statistics for Corpus Linguistics.* Edinburg University Press.

Prescher, Detlef. (2003).A Short Tutorial on the Expectation-Maximization Algorithm.In *Course Notes on CD of the 15th European Summer School in Logic, Language, And Information (ESSLLI-03).*

Schmid, H. (2000). Lopar: Design and Implementation. Arbeitspapiere des Sonderforschungsbereichs *340 Linguistic Theory and the Foundations of Computational Linguistics* 149, Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart.

Schulte im Walde, Sabine, Helmut Schmid, Mats Rooth, Stefan Riezler, Detlef Prescher. (2000). Statistical Grammar Models and Lexicon Acquisition. In *Linguistic Form and Its Computation.* CSLI Publications, Stanford GA.

Schulte im Walde, Sabine. (2000). The German Statistical Grammar Model: Development, Training, and Linguistic Exploitation.

Sproat, Richard. (1992). *Morphology and Computation*. MIT Press.

Roman Ondruљka, Jarmila Panevovб, and Jan Љtepбnek. (2003). An Exploitation of the Prague Dependency Treebank: A Valency Case*, Proceedings of The Shallow Processing of Large Corpora Workshop (SProLaC 2003).*